第5章

Pintos 的编译和运行

Pintos 操作系统运行在一个硬件仿真器 Bochs 上。该硬件仿真器以足够的精度模拟 80x86 CPU 以及它的外设，这样基于 80x86 CPU 的操作系统和软件未经修改就可以在其上运行。本教程使用的硬件仿真器为 Bochs。

本章介绍 Pintos 实验的运行环境和编译运行方法，以及如何调试 Pintos 内核。

5.1 实验环境

5.1.1 Pintos 运行环境说明

Pintos 操作系统可以直接运行在常规的采用 80x86 CPU 的 IBM 兼容 PC 上。但是为了实验和调试方便，**本教程以在 Linux 系统上通过 Bochs 仿真器运行 Pintos 为例介绍**。其中，Bochs 是一个系统仿真器，能够仿真 80x86 CPU 及其外设，从而所有能够在 80x86 CPU 上运行的操作系统和软件都可以不加修改地在 Bochs 上运行。

本教程采用的实验环境如图 5-1 所示，采用 Bochs 仿真器、GDB 调试工具，Pintos 运行在 Bochs 仿真器之上。

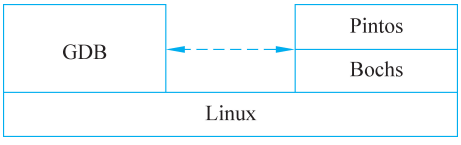

图 5-1 实验运行环境

5.1.2 实验环境说明

本教程提供的 vmware 虚拟机中已经安装和配置好全部的实验环境，包括 Ubuntu18 操作系统、Bochs 模拟器、Pintos 的源代码和编译环境、CGDB 调试器，以及可视化集成开发环境 Eclipse，并提供了一个已经配置好的 Eclipse 工程。该虚拟机默认 Ubuntu 用户名为 student，密码为 123456。打开主文件夹，可以看到上述文件已经被放置在该文件夹下，如图 5-2 所示。

其中 Eclipse 为可视化集成开发环境，Eclipse-workspace 为 Eclipse 的默认工

图 5-2　主文件夹

作空间文件夹，这两个文件夹的内容不必修改；Pintos 为 Pintos 的源代码，Pintos-bak 为 Pintos 源代码的备份文件，当需要恢复到初始状态时用该文件夹下的 src 目录替换 Pintos 文件夹下的 src 目录即可；Pintos-project2-only 为开展第二个 project 实验时需要的源代码文件；toolchain 为 Pintos 编译环境、Bochs 模拟器等文件的安装位置，该目录下的文件无须修改；“调试运行命令.txt”文件为事先已经写好的一份调试运行命令样例，便于大家使用。

5.2　Pintos 系统的编译与运行

根据实验内容的不同，需要在 src 的不同子目录下编译 Pintos 系统，以使生成的 Pintos 系统内核中包含所需的内容。相应地，对生成的 Pintos 系统的调试也最好在同一子目录下进行。以下以开展 Pintos 的第一个 project 实验（即 threads）所使用的编译方法为例进行介绍，后续实验的编译运行方法类似，只需要在不同的文件夹下运行编译命令（project 2 在 userprog 目录下，project 3 在 vm 目录下，project 4 在 filesys 目录下）即可。

5.2.1　编译

打开 Linux 系统的命令行接口 shell（后文简称终端），在其中通过命令行方式编译 Pintos 系统：通过 cd 命令进入 pintos/src/threads 目录，在该目录下执行 make 命令。如编译没有错误（编译信息没有报出 error），其结果如图 5-3 所示。

图 5-3　编译结果

该命令将会在本级目录下产生一个新的子目录 build，并在该子目录中生成一系列文件和子目录，如图 5-4 所示。

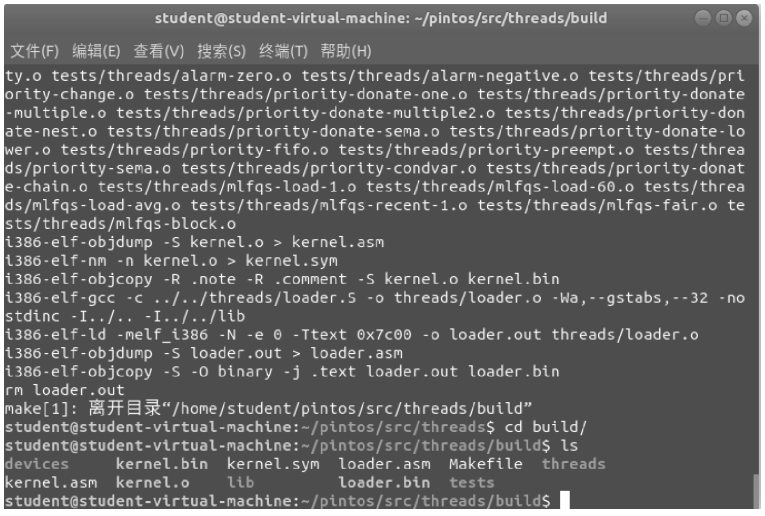

图 5-4 编译链接生成的文件

其中比较重要的文件如下所示。

- Makefile：这是 pintos/src/Makefile. build 文件的一个副本。该文件包含了 make 命令运行所需要的(关于编译 Pintos 内核的)相关信息。make clean 命令会重新创建该文件，因此，关于编译信息的永久性修改最好写入 pintos/src/Makefile. build 文件。
- kernel. o：是整个内核的目标文件，是所有内核源文件编译连接后形成的，包含调试信息，可以用 GDB 或 backtrace 来调试。
- kernel. bin：是内核的内存映像，是运行 Pintos 内核时加载到内存中的内容。
- loader. bin：是内核 boot loader 的内存映像。boot loader 用于将操作系统内核从磁盘读入内存，并跳转到操作系内核代码上去执行。该文件长度为 512 字节，是 PC BIOS 要求的长度。
- os. dsk：是内核的磁盘映像，实际由 loader. bin 和 kernel. bin 组成。仿真器将该文件视为虚拟磁盘。
- 其他子目录：包含目标文件(. o)和依赖(Dependency)文件(. d)，都是编译器产生的。其中依赖文件告诉 make 命令，当某些源文件或头文件被更改后，哪些源文件需要被重新编译。
- 若需要删除此前的编译结果，重新进行编译时，在 pintos/src/thread 目录下执行命令：

```
make clean
```

5.2.2 运行

为了方便 Pintos 系统的运行，在 src/utils/目录下提供了一个名为 Pintos 的 perl 程序

(注意区别于 Pintos 操作系统本身),该程序中包含了启动仿真器并在其上运行 Pintos 操作系统的一组命令。例如在 src/threads/下完成编译后执行命令:

```
pintos --bochs -v --run alarm-multiple
```

则 Pintos 程序会启动仿真器,在其上运行 Pintos 操作系统,并将 run alarm-multiple 作为参数传递给 Pintos 操作系统,运行结果如图 5-5 所示。

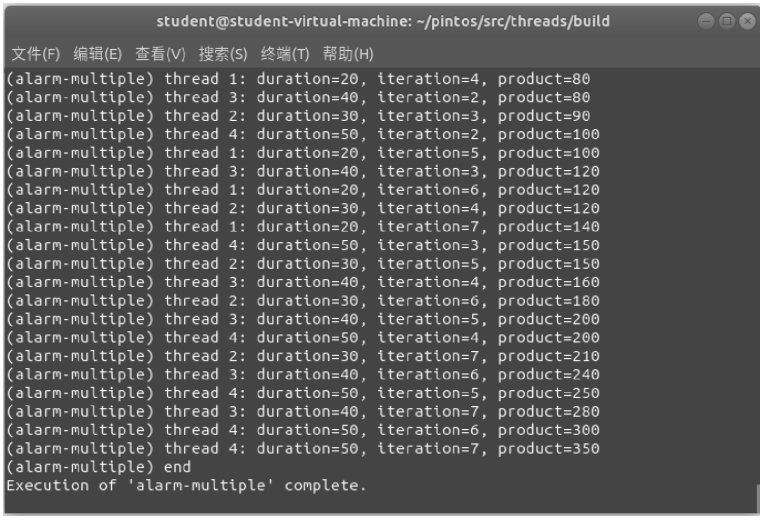

图 5-5　pintos --bochs -v -- run alarm-multiple 命令运行结果

其中 run 命令使 Pintos 操作系统在初始化完成后执行一个测试程序。alarm-multiple 便是该测试程序的名称。--bochs 选项表示使用 bochs 作为仿真器,-v 选项表示不打开该仿真器的主界面。

--bochs 和-v 这两个选项为可选内容,也就是说,实际运行时也可运行命令:

```
pintos --run alarm-multiple
```

表示使用默认的仿真器(默认为 bochs)并显示仿真器 GUI 主界面的模式下运行 alarm-multiple 测试程序。

查看参数或选项可以通过以下命令。

pintos:不带任何参数运行 Pintos 程序,将在终端上显示所有支持的选项;

pintos -h:在终端上显示 Pintos 操作系统支持的所有参数。

5.2.3　测试

为了检查读者所写代码的正确性,Pintos 系统为每个实验项目设计了一组测试程序。为了运行测试程序,可以进入 Pintos 安装目录下的 pintos/src/thread/build 目录中(后续其他实验分别需要进入相应的子目录),输入测试命令:

```
make check
```

每个测试程序运行后都会在终端上打印出测试成功或者失败的相关信息，如图 5-6 所示。其中，在测试失败时，给出被测程序产生的打印输出是否符合要求的相关信息。如图 5-7 所示，“+”表示被测程序产生的多余输出（测试程序根据打印输出来判断被测程序运行是否正确，因此在运行测试之前应当将测试程序中不必要的打印输出语句删除），“-”表示被测试程序应当产生却没有产生的输出。

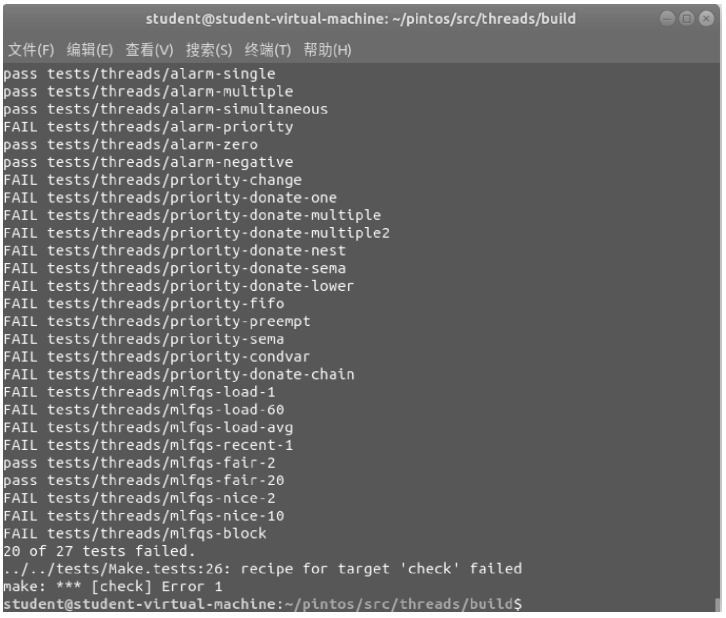

图 5-6　make check 测试结果

```
Acceptable output:
  (alarm-priority) begin
  (alarm-priority) Thread priority 30 woke up.
  (alarm-priority) Thread priority 29 woke up.
  (alarm-priority) Thread priority 28 woke up.
  (alarm-priority) Thread priority 27 woke up.
  (alarm-priority) Thread priority 26 woke up.
  (alarm-priority) Thread priority 25 woke up.
  (alarm-priority) Thread priority 24 woke up.
  (alarm-priority) Thread priority 23 woke up.
  (alarm-priority) Thread priority 22 woke up.
  (alarm-priority) Thread priority 21 woke up.
  (alarm-priority) end
Differences in `diff -u' format:
  (alarm-priority) begin
- (alarm-priority) Thread priority 30 woke up.
- (alarm-priority) Thread priority 29 woke up.
- (alarm-priority) Thread priority 28 woke up.
- (alarm-priority) Thread priority 27 woke up.
- (alarm-priority) Thread priority 26 woke up.
- (alarm-priority) Thread priority 25 woke up.
  (alarm-priority) Thread priority 24 woke up.
  (alarm-priority) Thread priority 23 woke up.
- (alarm-priority) Thread priority 22 woke up.
  (alarm-priority) Thread priority 21 woke up.
+ (alarm-priority) Thread priority 28 woke up.
+ (alarm-priority) Thread priority 25 woke up.
+ (alarm-priority) Thread priority 29 woke up.
+ (alarm-priority) Thread priority 22 woke up.
+ (alarm-priority) Thread priority 26 woke up.
+ (alarm-priority) Thread priority 30 woke up.
+ (alarm-priority) Thread priority 27 woke up.
  (alarm-priority) end
```

图 5-7　测试输出信息

同时，每个测试程序运行后也会在相关 build/tests/目录下产生一组与测试程序同名的文件(但扩展名不同)，分别为“[测试程序]. errors”“[测试程序]. output”“[测试程序]. result”，他们分别给出了测试程序运行后产生的错误信息、测试程序运行产生的完整输出，以及测试的结果。

5.3　Pintos 的调试

5.3.1　使用 GDB 调试

GNU 的调试器 GDB 是一款强大的调试工具，支持在源码级别查看程序的详细状态信息，单步查看每行代码的运行结果，查看变量值的变化以及内存的状态等。由于传统的 GDB 不具备代码同步展示功能，本教程在实验平台中安装了 CGDB 工具，其调试命令与 GDB 相同。

1. 运行调试

进入一个终端(如终端 1)，在目录 src/thread/build 中输入命令：

```
pintos --gdb -v --run alarm-multiple
```

如图 5-8 所示，终端 1 显示 Bochs 系统已经启动并等待 GDB 连接。接下来再打开一个终端(称为终端 2)，进入上述 src/thread/build 目录输入命令：

```
pintos-gdb kernel.o
```

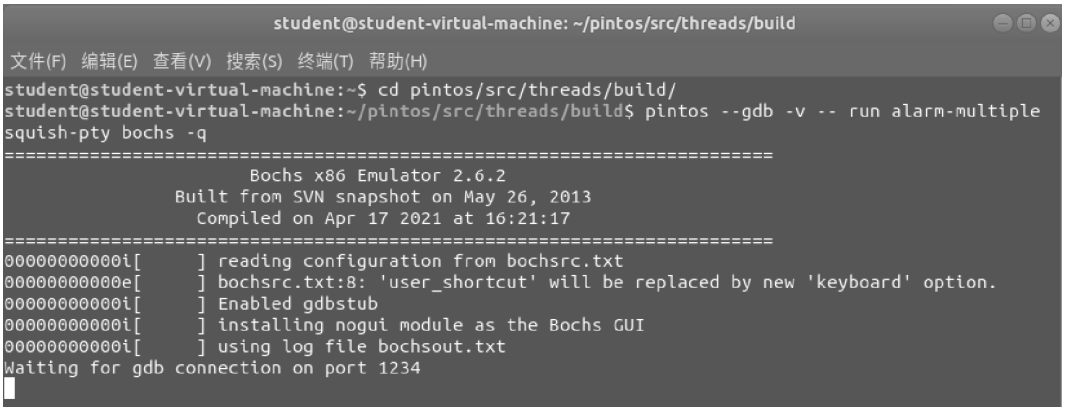

图 5-8　Pintos 系统等待 GDB 调试终端的连接

如图 5-9 所示，这样就进入了 CGDB。接下来在终端 2 中输入命令：

```
target remote localhost:1234
```

该命令也可以用 Pintos 设置好的宏命令 debugPintos 替换。

可看到终端 1 和终端 2 建立了连接。然后在终端 2 中输入命令：

```
c
```

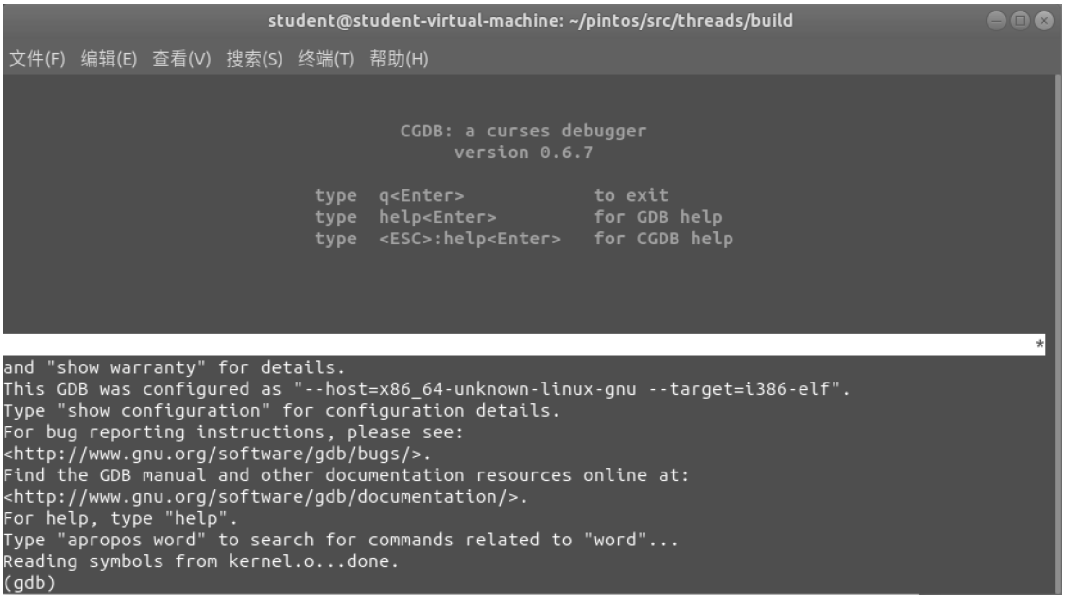

图 5-9 CGDB 调试环境

可在终端 1 中看到 Pintos 系统运行的信息。接下来通过在终端 2 中输入各种 GDB 调试命令，可以对终端 1 中 Pintos 系统的运行进行调试。

2. 调试命令

GDB 的调试命令有很多，常用的如表 5-1 所示。

表 5-1 GDB 调试命令

命　令	描　述	备　注
break(或 b)	设置断点	可以直接给出函数名，如 break main；也可以给出文件中的具体行号，如 break thread.c:23，也可以给出虚拟地址，如 break *0x80480e0 等
break line_num if condition	条件断点，当 condition 成立时在当前文件的 line_num 处设置断点	如 break 174 if (page==4)，表示当 page 为 4 时，在当前文件第 174 行处设置断点
condition brk_num expression	将已有断点 brk_num 设置为条件断点，其使能条件为 expression	
disable/ enable	暂停某个断点/继续使用某个断点	如 disable 1 表示暂停 1 号断点的使用；enable 1 表示继续开启 1 号断点
clear	清除断点	
continue(或 c)	连续运行至下一个断点	
next(或 n)	单步命令(不进入被调函数内部)	
step(或 s)	单步命令(进入被调函数内部)	
print(或 p)	显示表达式的值	如 p *thread_current()给出当前线程的结构

续表

命　令	描　述	备　注
display	每次命令执行后都显示的内容	如 display thread_current()->name，则每条 gdb 命令执行后都显示当前运行的线程名称
list	查看源代码	
info registers	查看整数寄存器当前的值	可以通过查看 cs 的最低两位来查看 CPU 当前执行代码的优先级别(privilege level)
info break	查看当前断点清单	
x/<n/f/u><addr>	查看 addr 内存地址开始后的 n 个单元的内容	关于该命令的详细介绍见下文
disassemble	反汇编	
quit(或者 q)	退出 GDB	
bt	查看调用栈帧	可以帮助分析函数调用顺序
file	装入预调试的可执行文件	如 file kernel.o
show	查看 GDB 的所有命令	
help	查看命令帮助信息	

3. 使用 Pintos 编写的 GDB 调试宏命令

Pintos 系统在 src/utils/目录下提供了一个名为 Pintos-gdb 的 shell 程序，该程序启动 GDB 并为之加载一些调试宏命令，这些调试宏命令有助于提高调试效率。宏命令见表 5-2。

表 5-2　宏命令

宏命令格式	功　能	备　注
debugPintos	是“target remote localhost：1234”的缩写	
dumplist & list type element	打印名为 list 的链表的表结点的内容	例如： dumplist & all_list thread allelem
btthreadlist list element	打印名为 list 的链表上所有线程的 backtrace	与 bt 命令相比，这条命令的优势在于可以打印非运行态的线程的 backtrace
btpagefault	在一个页错误(Page Fault)发生后，打印当前线程的 backtrace	

5.3.2　使用 Eclipse 调试

对于习惯用 GUI 环境进行调试的读者，本教程提供了配置好的 Eclipse 工程进行 Pintos 内核调试。该环境也使用 gdb 进行调试，只是通过 Eclipse 环境作为 gdb 的客户端，并将常用的调试命令通过图形化按钮实现。

打开主目录下的 Eclipse 文件夹，双击 Eclipse，打开的默认工程就是已经配置好的

Eclipse 工程，如图 5-10 所示。

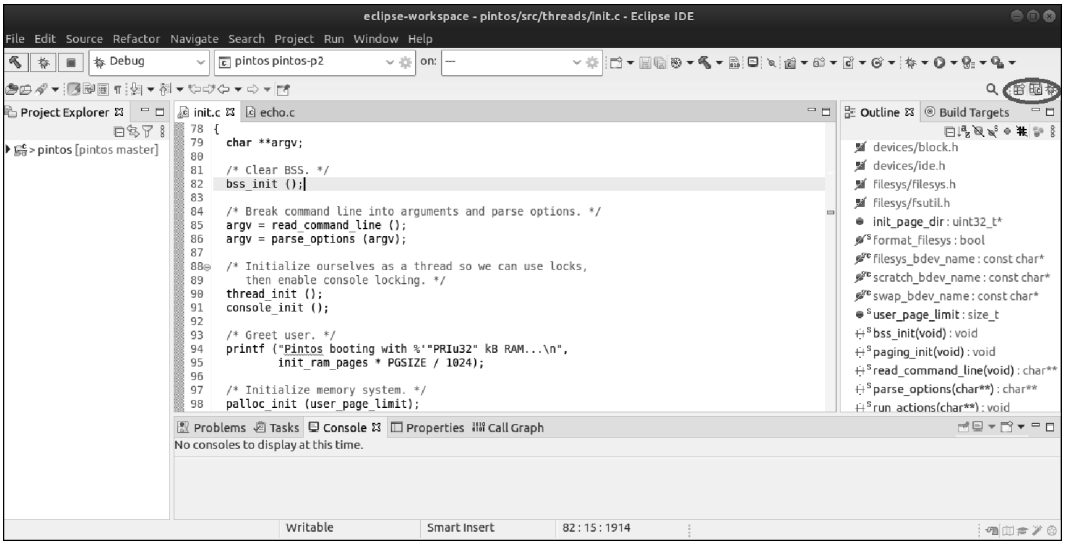

图 5-10　Eclipse 代码编辑与调试环境

可通过单击按钮(图 5-10 右上角圈出的按钮)在代码编辑模式与调试模式之间进行切换。调试方法如下。

(1) 启动调试服务器端：与启动 GDB 调试方法相同，进入一个终端(如终端 1)，在目录 src/thread/build 中输入命令：

```
pintos --gdb -v --run alarm-multiple
```

运行结果如图 5-11 所示。

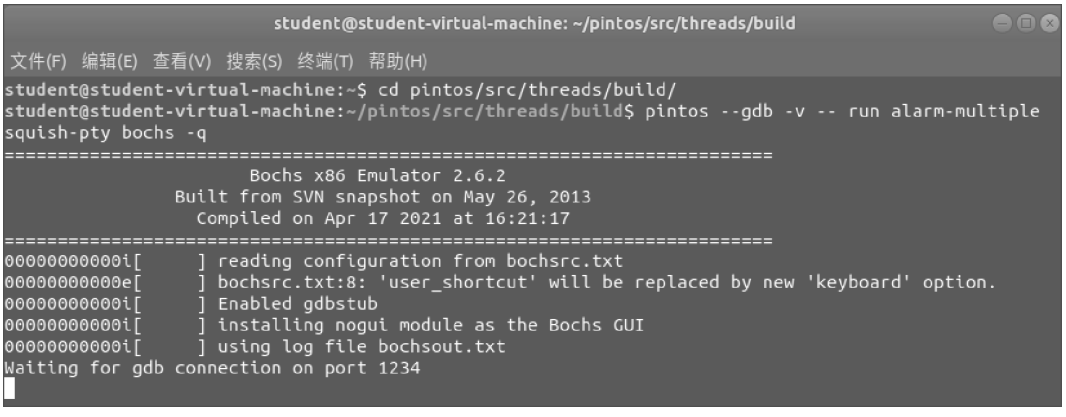

图 5-11　Pintos 系统等待 GDB 调试终端的连接

(2) 在 Eclipse 的工具栏中，单击 debug 按钮旁边的向下按钮，选择 Pintos Pintos-p1 选项，如图 5-12 所示，即可完成客户端连接(对于 project 2 需要选择 Pintos Pintos-p2)。

连接成功后，终端 1 中将打印出连接成功信息，并打印 Pintos 加载启动信息(图 5-13)，Eclipse 自动切换到调试模式下，在 debug 标签项中可以看到建立了新的调试任务，当前断